Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Improved binary similarity measures for software modularization*

Rashid NASEEM†1, Mustafa Bin Mat DERIS1, Onaiza MAQBOOL2, Jing-peng LI3, Sara SHAHZAD4, Habib SHAH5

(1*Faculty of Computer Science and Information Technology,Universiti Tun Hussein Onn Malaysia, Malaysia*)
(2*Department of Computer Science, Quaid-i-Azam University, Islamabad, Pakistan*)
(3*Division of Computer Science and Mathematics, University of Stirling, UK*)
(4*Department of Computer Science, University of Peshawar, Peshawar, Pakistan*)
(5*Faculty of Computer and Information Systems, Islamic University Madina, KSA*)
†E-mail: rnsqau@gmail.com

Received Oct. 30, 2015; Revision accepted Apr. 12, 2016; Crosschecked

Abstract: Various binary similarity measures have been employed in clustering approaches to make homogeneous groups of similar entities in the data. These similarity measures are mostly based only on the presence and absence of features. Binary similarity measures have also been explored with different clustering approaches (e.g., agglomerative hierarchical clustering) for software modularization to make the software systems understandable and manageable. Each similarity measure has its own strengths and weaknesses that result in improving and deteriorating the clustering results, respectively. This paper highlights the strengths of some well-known existing binary similarity measures for software modularization. Furthermore, based on these existing similarity measures, this paper introduces the improved new binary similarity measures. Proofs of the correctness with illustration and a series of experiments are presented to evaluate the effectiveness of our new binary similarity measures.

Key words: Binary similarity measure, Binary features, Combination of measures, Software modularization
http://dx.doi.org/10.1631/FITEE.1500373 **CLC number:**

1 Introduction

Clustering is an approach that makes clusters of similar entities in the data. Entities in a cluster are similar to each other (based on characteristics or features) while they are distinct from entities in other clusters. In the software domain, an important application of clustering is to modularize a software system or to recover the module architecture or components of the software systems by clustering the software entities, for example, functions, files or classes, in the source code. Recovery is very important when no up-to-date documentation of a software system is available (Shtern and Tzerpos, 2014). Besides clustering, other approaches have also been used for software modularization, for example, supervised clustering (Hall *et al.*, 2012), optimization techniques (Praditwong *et al.*, 2011), role-based recovery (Dugerdil and Jossi, 2008), graph-based techniques (Bittencourt and Guerrero, 2009), association-based approaches (Vasconcelos and Werner, 2007), spectral method (Xanthos and Goodwin, 2006), rough set theory (Jahnke, 2004), concept analysis (Tonella, 2001), and visualization tools (Synytskyy *et al.*, 2005).

* Project supported by the Office of Research, Innovation, Commercialization and Consultancy Office (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) (No. U062)
ORCID: Rashid NASEEM, http://orcid.org/0000-0002-4952-8100

A key activity in software clustering consists of gathering the entities from source code of software systems into meaningful and independent modules. The process of software clustering usually starts with the selection of entities and their features by parsing the source code of software systems. Then entities are organized into cohesive clusters by employing a particular clustering algorithm (Mitchell and Mancoridis, 2006).

Agglomerative Hierarchical Clustering (AHC) algorithms have been widely used by researchers to cluster the software systems (Muhammad *et al.*, 2012) (Maqbool and Babri, 2007) (Patel *et al.*, 2009) (Shtern and Tzerpos, 2010) (Mitchell, 2006) (Anquetil and Lethbridge, 1999) (Wiggerts, 1997). AHC comprises two main factors, a similarity measure to find the association between two entities and a linkage method to update the similarity values between entities in each iteration. However, selection of a similarity measure is an important factor in AHC (Cui and Chae, 2011) (Jackson *et al.*, 1989), which has a major influence on the clustering results (Naseem *et al.*, 2010) (Shtern and Tzerpos, 2012).

There exist a large number of binary similarity measures (Choi *et al.*, 2010) (Cheetham and Hazel, 1969). Nevertheless, for software modularization, the comparative studies have reported that Jaccard (JC) binary similarity measure produced better clustering results (Tzerpos and Holt, 2000) (Davey and Burd, 2000) (Lung *et al.*, 2004) (Shtern and Tzerpos, 2012). In our previous study (Naseem *et al.*, 2010), we proposed a new binary similarity measure, called JaccardNM (JNM), which could overcome some deficiencies of the JC binary similarity measure. We also examined the Russell&Rao (RR) binary similarity measure for software modularization for the first time, and found that it could generate better results as compared to JC and JNM binary similarity measures for some of the test software systems. In another study (Naseem *et al.*, 2013), we proposed COUSM (Cooperative Only Update Similarity Matrix) clustering algorithm, which combines two similarity measures in a single clustering process based on AHC.

In this paper, we explore the integration of the existing binary similarity measures for AHC algorithms using linkage methods (e.g., Complete Linkage (CL), Single Linkage (SL) and Weighted Average Linkage (WL) methods). For example, we select the JC similarity measure, which produces a relatively large number of clusters (Maqbool and Babri, 2004) (Saeed *et al.*, 2003) and the JNM binary similarity measure, which takes a less number of arbitrary decisions (Naseem *et al.*, 2010) during the clustering process. During the clustering process, creating a large number of clusters means that a clustering approach may create compact clusters, hence improving the quality of clustering results (Maqbool and Babri, 2007). Arbitrary decision is the arbitrary clustering of two entities when there exist more than two equally similar entities; hence, arbitrary decisions create problems and reduce the quality of clustering results (Naseem *et al.*, 2010) (Maqbool and Babri, 2007). This analysis leads us to introduce better binary similarity measures by combining the JC and JNM measures.

This study mainly focuses on the identification of the strengths of the existing binary similarity measures.Moreover, the improved similarity measures are based on the integration of JC, JNM, and RR similarity measures. While in our previous studies (Naseem *et al.*, 2010) (Naseem *et al.*, 2011), the main focus was to explore the deficiencies (i.e., creating a large number of equal similarity values and giving no importance to a pair of entities sharing a large number of features) of some well-known binary similarity measures and then solved these deficiencies by adding the total proportion of features to the denominator in the JC similarity measure.

The contributions of this paper can be summarized as:

1. Analysis of the Jaccard and JaccardNM (Naseem *et al.*, 2010) similarity measures for binary features and a comparison of their strengths;
2. The integration of the strengths of existing binary similarity measures to form new binary similarity measures for software clustering;
3. Introduction of four improved binary similarity measures that yield more effective solutions than the existing binary similarity measures.

   They are as follows:

   (a) *JCJNM*: add the JC and JNM binary similarity measures;

   (b) *JCRR*: add the JC and RR binary similarity measures;

(c) *JNMRR*: add the JNM and RR binary similarity measures;

(d) *JCJNMRR*: add the JC, JNM and RR binary similarity measures.

4. The additional evidence providing support to the correctness of the new binary similarity measures;

5. Conducting an experimental study that presents an external and internal evaluation of the proposed and existing similarity measures.

6. For all evaluation criteria, on average, our new similarity measures outperform the existing similarity measures.

The rest of this paper is organized as follows: Section 2 presents the preliminaries of software modularization using AHC algorithms. The strengths of the existing similarity measures are presented and illustrated in Section 3. This section also gives the derivations of our new similarity measures with proofs of correctness. Section 4 presents the experimental setup, which lists all the test software systems, similarity measures, linkage methods and evaluation criteria that are used to perform the series of experiments. Section 5 gives the experimental results and discussion on comparing our new similarity measures with existing similarity measures by using different external and internal criteria. Section 6 concludes this paper.

2 Software Modularization Using AHC

Clustering algorithms can be broadly categorized into hierarchical and partitional. As stated in Section 1, AHC has been commonly used for software modularization. AHC considers each entity to be a singleton cluster and groups the two most similar clusters at every step. At the end, it makes one large cluster, which contains all the entities, as shown in Algorithm 1.

Partitional clustering produces flat clusters with no hierarchy, and requires prior knowledge of the number of clusters. In the software domain, partitional clustering has also been used (Kanellopoulos *et al.*, 2007) (Lakhotia, 1997) (Shah *et al.*, 2013); however, there are some advantages of using AHC.

Algorithm 1 - Agglomerative Hierarchical Clustering (AHC) Algorithm

```
Input: Feature ($F$) matrix
Output: Hierarchy of Clusters (Dendrogram)
1: Create a similarity matrix by calculating similarity using
   a Similarity Measure between each pair of entities
2: repeat
3:     Group the most similar (singleton) clusters into one
       cluster (using maximum value of similarity in similarity
       matrix)
4:     Update the similarity matrix by recalculating similar-
       ity using a Linkage Method between newly formed
       cluster and existing (singleton) clusters
5: until the required number of clusters or a single large
   cluster is formed
```

For example, AHC does not require prior information about the number of clusters. Moreover, Wiggerts (1997) stated that the process of AHC is very similar to the approach of reverse engineering where architecture of a software system is recovered in a bottom-up fashion. AHC provides different levels of abstraction and can be useful for end users to select the desired number of clusters when the modularization results are meaningful to them (Lutellier *et al.*, 2015). Since a maintainer may not have the knowledge of the number of clusters in advance, therefore viewing the architecture at different abstraction levels facilitates understanding. Techniques have also been proposed to select an appropriate abstraction level, for example, (Chong *et al.*, 2013) proposed a dendrogram cutting approach for this purpose.

When AHC is utilized for software modularization, the first step that occurs is the selection of the entities to be clustered where each entity is described by different features. The steps are presented in more detail in the following subsections.

2.1 Selection of Entities and Features

Selecting the entities and features associated with entities depends on the type of software system and the desired architecture (e.g., layered/module architectures) to be recovered. For software modularization, researchers have used different types of entities, for example, methods (Saeed *et al.*, 2003), classes (Bauer and Trifu, 2004) and files (Andritsos and Tzerpos, 2005), (Anquetil and Lethbridge, 1999). Researchers have also used different types of features to describe the entities such as global variables used by an entity (Muhammad *et al.*, 2012), and procedure calls (Andritsos and Tzerpos, 2005).

Features are based on the relationships between entities, for example, containment and inheritance. Features may be in the binary or non-binary format. A binary feature represents the presence or absence of a relationship between two entities, while non-binary features are weighted features using different weighting schemes, for example, absolute and relative (Cui and Chae, 2011), to demonstrate the strength of the relationship between entities. Binary features are widely used in software modularization (Cui and Chae, 2011) (Mitchell and Mancoridis, 2006) (Wiggerts, 1997).

To apply AHC, a software system must be parsed to extract the selected entities and features associated with entities. This process results in a feature matrix of size *N x P*, where *N* is the total number of entities and *P* is the total number of features. Each entity in the feature matrix has a feature vector fi, fi = {f1, f2, f3, ..., fP}. More generally, *F* presents a general feature matrix, which takes values from $\{0,1\}^p$; in other words, $F = \{0,1\}^p$, where '1' means presence of a feature and '0' otherwise. AHC takes *F* as input, as shown in Algorithm I. Table 1 shows an example feature matrix *F* of a very small imaginary software system, which contains 5 entities (E1-E5) and 7 binary features (f1-f7). In Table 1, for example, f1 is present in entities E1, E2, and E3 while absent in entities E4 and E5.

Table 1 An Example Feature (*F*) Matrix

	f1	f2	f3	f4	f5	f6	f7
E1	1	1	0	0	0	0	0
E2	1	1	0	0	0	0	0
E3	1	0	1	1	0	0	0
E4	0	0	1	1	1	0	0
E5	0	0	0	0	0	1	0

2.2 Selection of Similarity Measure

The first step of the AHC process is to calculate the similarity between each pair of entities to obtain a similarity matrix by using a similarity measure, as shown in Step 1 of Algorithm I. Following are some well-known binary similarity measures for software modularization.

$$Jaccard(\mathbf{JC}) = \frac{a}{a+b+c} \tag{1}$$

$$JaccardNM(\mathbf{JNM}) = \frac{a}{2(a+b+c)+d} \tag{2}$$

$$Russell\&Rao(\mathbf{RR}) = \frac{a}{a+b+c+d} \tag{3}$$

All the existing binary similarity measures are expressed as functions of the following four quantities associated with the pair of entities (Ei, Ej), ∀ Ei, Ej ∈ *F* (Lesot and Rifqi, 2009):

- the number of features common to both entities, denoted by *a*
- the number of features present in Ei, but not in Ej, denoted by *b*
- the number of features present in Ej, but not in Ei, denoted by *c*
- the number of features absent in both entities, denoted by *d*.

It is important to note that $a+b+c+d$ is a constant value and is equal to the total number of features *P*. *a+b=0* occurs only when Ei has the feature vector fi = (0,...,0). Likewise, a+c=0 shows that Ej has feature vector fj = (0,...,0).

Definition 1 A binary similarity measure, *SM*, is a function whose domain is $\{0,1\}^p$, and whose range is $\mathbb{R}^+$, that is, $SM : \{0,1\}^p \rightarrow \mathbb{R}^+$ (Veal, 2011), with the following properties:

- Positivity: $SM(\text{Ei,Ej}) \geq 0$, ∀ Ei, Ej ∈ *F*
- Symmetry: $SM(\text{Ei,Ej}) = SM(\text{Ej,Ei})$, ∀ Ei, Ej ∈ *F*
- Maximality: $SM(\text{Ei,Ei}) \geq SM(\text{Ei,Ej})$, ∀ Ei, Ej ∈ *F*

Table 2 The similarity matrix derived from the matrix in Table 1 by using the JC similarity measure

	E1	E2	E3	E4	E5
E1					
E2	1				
E3	0.25	0.25			
E4	0	0	0.5		
E5	0	0	0	0.2	

To illustrate the calculation, for instance, of the JC measure as defined in Equation 1, Table 2 gives the similarity matrix of the feature matrix shown in Table 1. The similarity between E1 and E2 is calculated using the quantities defined by a, b, c, and d, and in this case a = 2, b = 0, c = 0, and d = 5. Putting all these values in JC similarity measure, we get similarity value '1' (shown in Table 2). Likewise, similarity values are calculated for each pair of entities and are presented in Table 2. Now AHC will group the most similar entities in Table 2, according to the Step 2 in Algorithm I. E1 and E2 have the highest similarity value, so AHC groups these entities in a single cluster (E1E2). A new cluster is therefore formed, and AHC will update the similarity values of E1E2 and all other (singleton) clusters, that is, E3, E4, and E5. To update these similarity values, different linkage methods can be used, which are described in the next subsection.

2.3 Selection of the Linkage Method

When a new cluster is formed, the similarities between new and the existing clusters are updated using a linkage method, as shown in step 3 of Algorithm I. There exist a number of methods, which update similarities differently. However, in this study, we discuss only those methods that are widely used for software modularization. They are listed below, where (EiEj) represents a new cluster and Ek represents an existing singleton cluster.

- Complete Linkage: *CL(EiEj, Ek)= min(sim(Ei, Ek), sim(Ej, Ek))*
- Single Linkage: *SL(EiEj, Ek)= max(sim(Ei, Ek), sim(Ej, Ek))*
- Weighted Average Linkage: *WL(EiEj, Ek)= 0.5 * (sim(Ei, Ek) + sim(Ej, Ek))*

In the illustrative example, we update similarity values between a new cluster (E1E2) and existing singleton clusters using CL method. The updated similarity matrix is shown in Table 3. For example, the CL method returns the minimum similarity value between E1 and E3 (0.25) and E2 and E3 (0.25). Both of the returned values are the same (if there was a minimum, then that value would be selected). Therefore, AHC selects this similarity value as the new similarity between (E1E2) and E3, as shown in Table 3. Similarly, all similarity values are updated between (E1E2) and E4 and E5.

AHC repeats Steps 2 and 3 until all entities are merged in one large cluster, or the desired number of clusters is obtained. At the end, AHC results in a hierarchy of clusters, also known as dendrogram, which is shown for the current example in Figure 1. The obtained hierarchy is then evaluated to assess the quality of the automatically formed clusters, and the performance of similarity measures and methods.

2.4 Assessment of the Results

Assessment of the clustering results is usually carried out using two approaches: external and internal assessment. The external assessment approach finds the association between automated results (decomposition) and the authoritative decomposition prepared by a human expert (e.g., original developer of the test software system). The approach is also known as *Authoritativeness*. The automated decomposition should resemble the authoritative decomposition as much as possible (Wu *et al.*, 2005). To find the authoritativeness, different measures may be used, such as, precision, recall (Sartipi and Kontogiannis, 2003), MoJo, and MoJoFM. Here, the widely used MoJoFM (Wen and Tzerpos, 2004) is discussed. MoJoFM is the updated version of the MoJo (Tzerpos and Holt, 1999) (Tzerpos, 2003), which calculates the move and join operations to convert the automated decomposition (M) into authoritative decomposition (N).

Table 3 The updated similarity matrix from the values in Table 2 using CL linkage method

	E1E2	E3	E4	E5
E1E2				
E3	0.25			
E4	0	0.5		
E5	0	0	0.2	

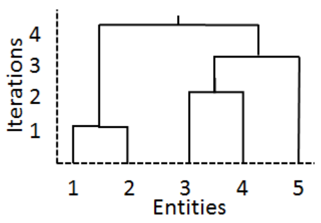

Fig. 1 Hierarchy created using the JC measure and CL method

$$MoJoFM(M,N) = \left(1 - \frac{mno(M,N)}{max(mno(\forall M,N))}\right) * 100 \quad (4)$$

where $mno(M,N)$ is the minimum number of 'move' and 'join' operations required to translate M in to N and $max(mno(\forall M,N))$ is the maximum of $mno(\forall M,N)$. MoJoFM produces percentage of the similarity between two decompositions. A higher percentage shows greater correspondence between the two decompositions and hence better results, while lower percentage indicates that the decompositions are different.

Internal assessment is to evaluate the quality of the internal characteristics of the clusters in automated decomposition. There exist a number of measures to evaluate the cluster quality internally, for example, arbitrary decisions (Wang *et al.*, 2010), number of clusters (Wang *et al.*, 2010), size of clusters (extremity) (Glorie *et al.*, 2009), modularization quality (Praditwong, 2011), and coupling and cohesion (Cui and Chae, 2011). In this study, we intend to use arbitrary decisions and number of clusters. Arbitrary decisions are taken by AHC when there exist more than one maximum similarity values in the similarity matrix during iteration. Thus, the decision of selecting the maximum value is arbitrary, since more than one pair of entities is equally similar. A large number of arbitrary decisions shows poor characteristic of the clusters, while a less number of arbitrary decisions mean good characteristic, in terms of authoritativeness (Naseem *et al.*, 2013). The number of clusters is another internal assessment criterion that is used to evaluate cluster quality. If an AHC produces a large number of clusters during the clustering process, it means that clusters are compact and have good quality (Maqbool and Babri, 2007). A less number of clusters during clustering means that they are large in size and less compact, and hence are considered poor quality.

3 New Similarity Measures

As discussed in Section 1, we define new similarity measures that have the combined strengths of existing similarity measures JC, JNM, and RR defined in Equations 1, 2 and 3, respectively. These three similarity measures have shown better results for software modularization as compared to other measures (Cui and Chae, 2011). To highlight the strengths of these existing measures, we first present a small example case study, and then define our new similarity measures.

3.1 An Example Case Study

To illustrate the strengths of existing similarity measures, we take a small example of an imaginary feature matrix from (Naseem *et al.*, 2013). The example feature matrix is shown in Table 4, with 8 entities (E1–E8) and 13 features (f1–f13). Using feature matrix shown in Table 4, we illustrate the strengths of JC and JNM similarity measures. We use the CL method in AHC using the CL with JC and the CL with JNM on the feature matrix in Table 4.

Table 4 An Example Feature Matrix

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13
E1	0	0	0	0	0	1	1	1	1	1	1	1	1
E2	0	0	0	0	0	1	1	1	1	1	1	1	1
E3	1	1	0	0	0	0	0	0	0	0	0	0	0
E4	1	1	0	0	0	0	0	0	0	0	0	0	0
E5	1	1	1	1	1	0	0	0	0	0	0	0	0
E6	1	1	1	1	0	0	0	0	0	0	0	0	0
E7	0	0	1	1	1	1	1	0	0	1	0	1	0
E8	0	0	1	1	1	1	0	1	1	0	1	0	0

3.1.1 JC with CL clustering process

First, we illustrate the JC measure with the CL method. The first step of AHC is to create the similarity matrix using a similarity measure. After applying the JC measure to feature matrix in Table 4, we get the similarity matrix shown in Table 5. In the first iteration of AHC, a maximum similarity value from the similarity matrix (Table 5) is selected to make a new cluster or update a cluster. So, AHC searches for a maximum similarity value in Table 5 but it finds maximum similarity value '1' twice (i.e., for (E1E2) and (E3E4)). AHC may select either, but we enforced AHC to select the last occurring value, that is, similarity value of (E3E4) (see Table 6).

The CL method is used to update the similarity values between the new cluster (E3E4) and all existing singleton clusters, and the updated similarity matrix is shown in Table 6. In the second iteration, AHC searches again for the maximum value in updated similarity matrix Table 6. This time it makes (E1E2) as a new cluster and updates its similarity values with all other existing clusters, as shown in Table 7. In iterations 3 and 4, it makes clusters of (E5E6) and (E7E8), as shown in Table 8 and Table

9, respectively. From Table 8, it can be seen that there are two maximum values (0.4), hence AHC may select either again. As stated earlier, AHC will select a value that occurs later; therefore, it a makes cluster (E7E8). In the remaining iterations, AHC makes clusters of ((E3E4)(E5E6)), ((E1E2)(E7E8)) and (((E1E2) (E7E8)) ((E3E4) (E5E6))), as shown in Tables 10–12.

Table 5 Similarity matrix of table 4 using the Jaccard (JC) similarity measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**1**							
E3	0	0						
E4	0	0	**1**					
E5	0	0	0.4	0.4				
E6	0	0	0.5	0.5	0.8			
E7	0.363	0.363	0	0	0.333	0.222		
E8	0.363	0.363	0	0	0.333	0.222	0.4	

Table 6 Iteration 1: updated similarity matrix of Table 5 using the CL method

	E1	E2	(E3E4)	E5	E6	E7	E8
E1							
E2	**1**						
(E3E4)	0	0					
E5	0	0	0.4				
E6	0	0	0.5	0.8			
E7	0.363	0.363	0	0.333	0.222		
E8	0.363	0.363	0	0.333	0.222	0.4	

Table 7 Iteration 2: updated similarity matrix of Table 6 using the CL method

	(E1E2)	(E3E4)	E5	E6	E7	E8
(E1E2)						
(E3E4)	0					
E5	0	0.4				
E6	0	0.5	**0.8**			
E7	0.363	0	0.333	0.222		
E8	0.363	0	0.33	0.222	0.4	

3.1.2 JNM with the CL clustering process

Now we apply the JNM measure on the feature matrix given in Table 4, and get a similarity matrix that can be seen in Table 13. The process for making clusters is the same as discussed in Subsection 3.1.1. As per the AHC, the first cluster formed is (E1E2), second is (E5E6), third is (E7E8), fourth is ((E1E2) (E7E8)), fifth is (E3E4), sixth is ((E3E4) (E5E6)), and the last is (((E1E2) (E7E8)) ((E3E4) (E5E6))). The similarity matrices during iterations—from the first iteration to the seventh (n-1) iteration—are given in Tables 14 to 20. In each

Table 8 Iteration 3: Updated Similarity Matrix of Table 7 Using the CL Method

	(E1E2)	(E3E4)	(E5E6)	E7	E8
(E1E2)					
(E3E4)	0				
(E5E6)	0	**0.4**			
E7	0.363	0	0.222		
E8	0.363	0	0.222	**0.4**	

iteration, the CL method is used to update the similarity between newly formed and existing (singleton) clusters.

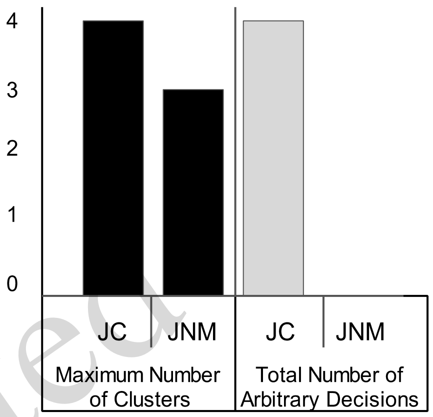

Fig. 2 Number of clusters and arbitrary decisions created by JC and JNM

Table 9 Iteration 4: Updated Similarity Matrix of Table 8 Using the CL Method

	(E1E2)	(E3E4)	(E5E6)	(E7E8)
(E1E2)				
(E3E4)	0			
(E5E6)	0	**0.4**		
(E7E8)	0.363	0	0.2	

Table 10 Iteration 5: Updated Similarity Matrix of Table 9 Using the CL Method

	(E1E2)	((E3E4)(E5E6))	(E7E8)
(E1E2)			
((E3E4)(E5E6))	0		
(E7E8)	**0.363**	0	

Table 11 Iteration 6: Updated Similarity Matrix of Table 10 Using the CL Method

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	0	

Table 12 Iteration 7: Updated Similarity Matrix of Table 11 Using the CL Method

	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

Table 13 Similarity Matrix of Feature Matrix in Table 4 Using the JNM Similarity Measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**0.381**							
E3	0	0						
E4	0	0	0.133					
E5	0	0	0.111	0.111				
E6	0	0	0.118	0.118	0.222			
E7	0.166	0.166	0	0	0.136	0.09		
E8	0.166	0.166	0	0	0.136	0.09	0.173	

Table 14 Iteration 1: Updated Similarity Matrix of Table 13 Using the CL Method

	(E1E2)	E3	E4	E5	E6	E7	E8
(E1E2)							
E3	0						
E4	0	0.133					
E5	0	0.111	0.111				
E6	0	0.118	0.118	**0.222**			
E7	0.166	0	0	0.136	0.09		
E8	0.166	0	0	0.136	0.09	0.173	

Table 15 Iteration 2: Updated Similarity Matrix of Table 14 Using the CL Method

	(E1E2)	E3	E4	(E5E6)	E7	E8
(E1E2)						
E3	0					
E4	0	0.133				
(E5E6)	0	0.111	0.111			
E7	0.166	0	0	0.09		
E8	0.166	0	0	0.09	**0.173**	

Table 16 Iteration 3: Updated Similarity Matrix of Table 15 Using the CL Method

	(E1E2)	E3	E4	(E5E6)	(E7E8)
(E1E2)					
E3	0				
E4	0	0.133			
(E5E6)	0	0.111	0.111		
(E7E8)	**0.166**	0	0	0.09	

Table 17 Iteration 4: Updated Similarity Matrix of Table 16 Using the CL Method

	((E1E2)(E7E8))	E3	E4	(E5E6)
((E1E2)(E7E8))				
E3	0			
E4	0	**0.133**		
(E5E6)	0	0.111	0.111	

Table 18 Iteration 5: Updated Similarity Matrix of Table 17 Using the CL Method

	((E1E2)(E7E8))	(E3E4)	(E5E6)
((E1E2)(E7E8))			
(E3E4)	0		
(E5E6)	0	**0.111**	

Table 19 Iteration 6: Updated Similarity Matrix of Table 18 Using the CL Method

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	**0**	

Table 20 Iteration 7: Updated Similarity Matrix of Table 19 Using the CL Method

	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

3.1.3 Discussion on the results of JC and JNM measures

In the previous two Subsections 3.1.1 and 3.1.2, we observed that the JC measure results in more clusters during clustering as compared to the JNM measure. Figure 2 shows the maximum number of clusters achieved and the total number of arbitrary decisions made during the clustering process for both measures. It can be seen from Figure 2 that the JNM measure produces a less number of arbitrary decisions as compared to the JC measure. The JNM produces results as expected because the main intuition of introducing this measure is to reduce the arbitrary decisions (Naseem *et al.*, 2010). Hence, from these results we can easily conclude that the JC has the strength to create more clusters during the clustering process, while the JNM has the strength to reduce the number of arbitrary decisions.

It has been shown that if a clustering approach results in a large number of clusters during the clustering process, it increases the cohesiveness of the clusters and hence increases the authoritativeness of the results (Maqbool and Babri, 2007). Second, in our previous study, we showed that if a clustering approach reduces arbitrary decisions, it would create more authoritative results (Naseem *et al.*, 2013). This analysis leads us to define the new measures that may have the characteristics of these existing similarity measures. It would be useful to integrate the existing similarity measures and come up with the new measures to increase the number of clusters and reduce the arbitrary decisions during the clustering process.

3.2 The New Binary Similarity Measures

According to the aforementioned discussion, the JC measure has the strength of creating a large number of clusters during the clustering process, while the JNM measure has the strength of creating clusters with a less number of arbitrary decisions. We also consider the RR measure because for some case studies, it has produced better clustering results for software modularization (Naseem *et al.*, 2010). RR reduces the arbitrary decisions when JC creates, for example, when the values of a among the entities are different and $b + c = 0$. To combine the strengths of these existing similarity measures, the add operation is used to combine the existing similarity measures. We introduce four new similarity measures as follows:

3.2.1 Addition of the JC and JNM measures: "JCJNM" similarity measure

The strengths of similarity measures shown in Subsection 3.1 can be combined by adding both the similarity measures (JC and JNM) to obtain the "JCJNM" binary similarity measure. Our new measure"JCJNM" is defined as:

$$JCJNM = JC + JNM$$

$$= \frac{a}{s} + \frac{a}{2s + d}$$

$$= \frac{a(3s + d)}{s(2s + d)} \qquad (5)$$

where

$$s = a + b + c$$

3.2.2 The Example Case Study and JCJNM Measure

To demonstrate the strengths of our new measure, we now apply the JCJNM similarity measure to the example feature matrix shown in Table 4. The corresponding similarity matrix using the JCJNM similarity measure is shown in Table 21. The CL method is used to update the similarity matrix during the clustering process. We can see from Table 21 that the JCJNM prioritizes the similarity values between the pair of entities (E1E2) and (E3E4), as done by the JNM in the similarity matrix given in Table 13. Hence, the decision to cluster the entities is no longer arbitrary. Entities E1 and E2 have a high value of similarity and are grouped first (see Table 22). Then in the subsequent iterations, the AHC makes clusters of (E3E4), (E5E6) and (E7E8). Note that in Iteration 3 as given in Table 8, the JC measure creates arbitrary decisions while our new measure JCJNM does not, as shown in Table 24.

It is interesting to note that the JCJNM measure creates clusters as created by the JC measure (4 clusters) and similar to the JNM measure, makes no arbitrary decisions. It can be inferred that our new measure has the strength to create a large number of clusters while reducing the arbitrary decisions made by the AHC during the clustering process. Therefore, the JCJNM outperforms the existing similarity measure.

Table 21 Similarity Matrix of Feature Matrix in Table 4 Using the JCJNM Similarity Measure

	E1	E2	E3	E4	E5	E6	E7	E8
E1								
E2	**1.381**							
E3	0	0						
E4	0	0	1.133					
E5	0	0	0.511	0.511				
E6	0	0	0.618	0.618	1.022			
E7	0.526	0.526	0	0	0.466	0.31		
E8	0.526	0.526	0	0	0.466	0.31	0.573	

Table 22 Iteration 1: Updated Similarity Matrix of Table 21 Using the CL Method

	(E1E2)	E3	E4	E5	E6	E7	E8
(E1E2)							
E3	0						
E4	0	**1.133**					
E5	0	0.511	0.511				
E6	0	0.618	0.618	1.022			
E7	0.526	0	0	0.466	0.31		
E8	0.526	0	0	0.466	0.31	0.573	

Table 23 Iteration 2: Updated Similarity Matrix of Table 22 Using the CL Method

	(E1E2)	(E3E4)	E5	E6	E7	E8
(E1E2)						
E3E4	0					
E5	0	0.511				
E6	0	0.618	**1.022**			
E7	0.526	0	0.466	0.31		
E8	0.526	0	0.466	0.31	0.573	

Table 24 Iteration 3: Updated Similarity Matrix of Table 23 Using the CL Method

	(E1E2)	(E3E4)	(E5E6)	E7	E8
(E1E2)					
(E3E4)	0				
(E5E6)	0	0.511			
E7	0.526	0	0.31		
E8	0.526	0	0.31	**0.573**	

Table 25 Iteration 4: Updated Similarity Matrix of Table 24 Using the CL Method

	(E1E2)	(E3E4)	(E5E6)	(E7E8)
(E1E2)				
(E3E4)	0			
(E5E6)	0	0.511		
(E7E8)	**0.526**	0	0.31	

Table 26 Iteration 5: Updated Similarity Matrix of Table 25 Using the CL Method

	((E1E2)(E7E8))	(E3E4)	(E5E6)
((E1E2)(E7E8))			
(E3E4)	0		
(E5E6)	0	**0.511**	

Table 27 Iteration 6: Updated Similarity Matrix of Table 26 Using the CL Method

	((E1E2)(E7E8))	((E3E4)(E5E6))
((E1E2)(E7E8))		
((E3E4)(E5E6))	0	

Table 28 Iteration 7: Updated Similarity Matrix of Table 27 Using the CL Method

	(((E1E2)(E7E8))((E3E4)(E5E6)))
(((E1E2)(E7E8))((E3E4)(E5E6)))	

Proposition 3.1 Let the range of JCJNM be z (P represents the total number of features), then

$$JCJNM(Ei,Ej) = \begin{cases} z = 1.5, & \text{if } a = P \\ 0 < z < 1.5, & \text{if } 0 < a < P \\ z = 0, & \text{if } a = 0 \end{cases}$$

Proof. JCJNM is the combined function of four quantities a, b, c, and d, as shown in Equation 5. Substituting value of $s = a + b + c$.

$$JCJNM(Ei,Ej) = \frac{a(3(a+b+c)+d)}{(a+b+c)(2(a+b+c)+d)} \quad (6)$$

If all features are present in feature vectors of Ei and Ej, that is, $b = c = d = 0$, and $a =$ P, then the above equation reduces to

$$JCJNM(Ei,Ej) = \frac{a(3(a))}{(a)(2(a))} = 1.5 \quad (7)$$

Therefore, the maximum similarity value that JCJNM can produce is 1.5
Now, if no common feature is present in feature vectors of Ei and Ej, that is, $a = 0$ and $b + c + d \geq 0$, then Equation 6 reduces to

$$JCJNM(Ei,Ej) = \frac{0(3(0+b+c)+d)}{(0+b+c)(2(0+b+c)+d)} = 0 \quad (8)$$

Thus the minimum similarity value that JCJNM can produce is 0.

Lastly, if there exist some common and absent features in feature vectors of Ei and Ej, that is, if $a = x$ and $b = c = d = y$, where x, $y > 0$, then Equation 5 reduces to

$$JCJNM(Ei,Ej) = \frac{x(3(x+y+y)+y)}{(x+y+y)(2(x+y+y)+y)}$$

The above equation simplifies to

$$JCJNM(Ei,Ej) = \frac{3x^2+7xy}{2x^2+9xy+10y^2} \quad (9)$$

Equation 9 results in values between 0 and 1.5, if x, y>0, $\forall$ Ei, Ej $\in F$.

□

Proposition 3.2 JCJNM satisfies Definition 2.1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^p$ and range is $\mathbb{R}^+$.

Proof. JCJNM is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated only using binary values in feature vector of entities as defined in Section 2.2. Hence, the domain of JCJNM measure is$\{0,1\}^p$. Meanwhile, JCJNM results in a real value, that is, $z = \mathbb{R}^+$ as proved in Proposition 3.1. □

Proposition 3.3 JCJNM fulfills the properties of Positivity and Symmetry

Proof. Positivity: It has been shown in the proof of Proposition 3.1, that JCJNM creates similarity value equal to or greater than 0, that is, JCJNM(Ei,Ej) $\rightarrow \mathbb{R}^+$, $\forall$ Ei, Ej $\in F$.

Symmetry: JCJNM is the combined function of four quantities a, b, c, and d, and all these quantities are symmetric, so it is obvious that

$$JCJNM(Ei,Ej) = JCJNM(Ej,Ei)$$

□

Proposition 3.4 JCJNM fulfills the property of Maximality of similarity measure.

Proof. Let us suppose that $b + c = x$ and x is a positive number, then Equation 5 becomes

$$JCJNM(Ei,Ej) = \frac{a(3(a+x)+d)}{(a+x)(2(a+x)+d)}$$

The above equation simplifies to

$$JCJNM(Ei,Ej) = \frac{a(3a+d+3x)}{a(2a+d+2x)} \quad (10)$$

To calculate the similarity of an entity with itself, that is, JCJNM(Ei,Ei) then $x = 0$,a, $d \geq 0$. Using these quantities, Equation 10 of JCJNM reduces to:

$$JCJNM(Ei,Ei) = \frac{a(3a+d)}{a(2a+d)} \quad (11)$$

Therefore, using Equations 11 and 10, $\forall$ Ei, Ej $\in F$, the following association will always be true for

$a + d \geq 1$ and x, where $a + d + x = P$:[1]

$$\frac{a(3a+d)}{a(2a+d)} > \frac{a(3a+d+x)}{a(2a+d+x)} \qquad if(x>0)$$

$$\geq \frac{a(3a+d+3x)}{a(2a+d+2x)} \qquad if(x \geq 0)$$

□

3.2.3 Addition of the JC and RR measures: "JCRR" similarity measure

The second new similarity measure that we derive is "JCRR". This measure adds RR similarity measure with JC similarity measure. The derivation of JCRR is given as:

$$JCRR = JC + RR$$

$$= \frac{a}{s} + \frac{a}{s+d}$$

$$= \frac{a(2s+d)}{s(s+d)} \tag{12}$$

For proofs of the correctness of JCRR binary similarity measure, please refer to Appendix A.

3.2.4 Combination of the JNM and RR measures: "JNMRR" similarity measure

The third combination of the existing similarity measure is JNM and RR. We add both the similarity measures and the derived similarity measure is called "JNMRR" and defined as:

$$JNMRR = JNM + RR$$

$$= \frac{a}{2s+d} + \frac{a}{s+d}$$

$$= \frac{a(3s+2d)}{(2s+d)(s+d)} \tag{13}$$

For proofs of the correctness of JNMRR binary similarity measure, please refer to Appendix B.

[1]We have two associations, that is, equality and maximality. 1) Equality: for instance, let $x = 0$, then Equation 10 becomes equal to Equation 11. 2) Maximality: let a = d = x = 1, so a + b + x = P = 3, then the association between Equations 11 and 10 becomes

$$\frac{(3+1)}{(2+1)} > \frac{(3+1+3)}{(2+1+2)}$$

3.2.5 Addition of the JC and JNM and RR measures: "JCJNMRR" similarity measure

Finally, we add all the three existing measures and come up with a new measure "JCJNMRR". This measure adds the JC with JNM, with RR as follows:

$$JCJNMRR = JC + JNM + RR$$

$$= \frac{a}{s} + \frac{a}{2s+d} + \frac{a}{s+d}$$

$$= \frac{a(5s^2+5sd+d^2)}{s(2s^2+3sd+d^2)} \tag{14}$$

For proofs of the correctness of JNMRR binary similarity measure, please refer to Appendix C.

4 The Experimental Setup

AHC produces different results due to the biases of the different linkage methods and similarity measures (Cui and Chae, 2011). We have performed a number of experiments by employing well known basic linkage methods using existing and our new similarity measures. In this section, we discuss the test systems used for experimental purposes and the clustering process setup including the selection of assessment criteria.

4.1 Test Systems

We have used 8 test systems that are developed using C, C++, and Java languages to conduct the experiments. The test systems have been used in previous research (Siddique and Maqbool, 2012), (Muhammad *et al.*, 2012), (Naseem *et al.*, 2013). All of these test systems vary in their source code sizes and application domains. Table 29 presents the details of the test software systems.

Table 29 Details of the Test Software Systems

ID	Test System	Number of Classes	LOC	Entity
1	DDA	90	82877	Class
2	FES	47	10402	Class
3	Mozilla	1202 Files	400000	File
4	PEDS	41	16360	Class
5	PLC	69	51768	Class
6	PLP	72	50661	Class
7	SAVT	97	27311	Class
8	Weka	331	100000	Class

We used two open source and six proprietary test software systems. The open source test software systems are: 1) Mozilla, an open source web browser and developed in C and C++ programming languages. For experiments, we used Mozilla[2] version 1.3 released in March 2003. This test system is taken from Siddique and Maqbool (2012); 2) Weka, another open source software system developed in Java programming language, is a well-known data mining software system used for data pre-processing, clustering, regression, classification, association rules, and visualization. We use Weka[3] version 3.4, taken from Siddique and Maqbool (2012), for experimental purposes.

The proprietary test software systems used for experiments are developed in Visual C++ programming language. They are: 1) DDA, software to design the document composition and layout; 2) FES, a fact extractor software system to extract the facts of software systems; 3) PEDS, a power dispatch problem solver using conventional and evolutionary computing techniques; 4) PLC, a printer language converter software system used to convert the intermediate data structure to a well-known printer language; 5) PLP, a parser software system, which is used to parse a well-known printer language; 6) SAVT, a statistical and analysis visualization tool. These software systems are proprietary and are currently operational. We obtained the extracted feature matrix from Muhammad *et al.* (2012).

4.2 Entities and Features

Mozilla's data set is taken from Siddique and Maqbool (2012), who considered files as entities because a .c or .cpp file contains both functions with and without classes. Hence, in this study, files are considered entities for Mozilla, and file calling is used as a feature. The total number of file calling feature is 258. For Weka test system, which is purely developed Java, we consider classes to be entities (Siddique and Maqbool, 2012). Since classes are considered to be the basic building blocks of object oriented languages. Siddique and Maqbool (2012) selected functions invoked, user-defined types and global variables as features for classes.

For the proprietary software systems, Muhammad *et al.* (2012) considered classes to be entities. We select ten indirect features for these entities (shown in Table 30), since indirect features give better results as compared to direct features (Muhammad *et al.*, 2012). We consider various types of test software systems that are developed in different programming languages, with different types of entities and features because we want to see whether our new proposed similarity measures are applicable to these different artifacts to make good clusters for software modularization.

Table 30 Indirect Features of the Classes in Proprietary Test Systems That were Used for Experiments

Feature Type	DDA	FES	PEDS	PLC	PLP	SAVT
Same Inheritance Hierarchy	98	166	70	26	64	986
Same Class Containment	58	56	12	58	144	1032
Same Class in Methods	476	384	76	162	672	1900
Same Generic Class	59	91	6	465	98	49
Same Generic Parameter	0	4	0	0	0	0
Same File	136	42	36	1812	826	264
Same Folder	2456	0	0	0	0	0
Same Macro Access	0	0	12	0	2	0
Same Global Access	918	0	0	0	268	0
Total Features	4201	743	212	2523	2074	4231

4.3 Clustering Strategies

In this study, we have considered existing similarity measures and linkage methods that have produced better results for software modularization (Muhammad *et al.*, 2012) (Davey and Burd, 2000). In order to conduct the experiments, we categorize clustering actors into different clustering stratagems shown in Table 31. These stratagems are composed of three existing similarity measures and four new similarity measures, using three well-known basic linkage methods. The details of these stratagems are given in Table 31.

[2]ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/releases/mozilla1.3/src/

[3]http://perun.pmf.uns.ac.rs/radovanovic/dmsem/cd/install/Weka/doc/html/Weka%203.4.5.htm

Table 31 Clustering Stratagems

Sr. No.	SM	Abbr	Linkage Method	Abbr
1	JCJNM	JCJNM	Complete	CL
2	JCRR	JCRR	Complete	CL
3	JNMRR	JNMRR	Complete	CL
4	JCJNMRR	JCJNMRR	Complete	CL
5	JCJNM	JCJNM	Single	SL
6	JCRR	JCRR	Single	SL
7	JNMRR	JNMRR	Single	SL
8	JCJNMRR	JCJNMRR	Single	SL
9	JCJNM	JCJNM	Weighted Average	WL
10	JCRR	JCRR	Weighted Average	WL
11	JNMRR	JNMRR	Weighted Average	WL
12	JCJNMRR	JCJNMRR	Weighted Average	WL
13	Jaccard	JC	Complete	CL
14	JaccardNM	JNM	Complete	CL
15	Russell&Rao	RR	Complete	CL
16	Jaccard	JC	Single	SL
17	JaccardNM	JNM	Single	SL
18	Russell&Rao	RR	Single	SL
19	Jaccard	JC	Weighted Average	WL
20	JaccardNM	JNM	Weighted Average	WL
21	Russell&Rao	RR	Weighted Average	WL

4.4 Assessment Criteria

To assess the output of the stratagems given in Subsection 4.3, we considered external as well as internal assessment. External assessment is the approach in which expert decomposition is required to evaluate the automated results, also known as authoritativeness. As AHC produces results at each iteration, the following question arises: which iteration's result should be evaluated? To answer the question, researchers used external criterion to evaluate results of each iteration, and then presented the maximum or average value of the criterion (Wen and Tzerpos, 2004), (Muhammad *et al.*, 2012), (Maqbool and Babri, 2007). We report the results by selecting the maximum MoJoFM value out of all values obtained from each iteration. To show the strengths and weakness of the proposed and existing measures, we also evaluate the experimental results using internal assessment criteria, that is, arbitrary decisions (Naseem *et al.*, 2013) and the number of clusters produced during clustering (Maqbool and Babri, 2007).

4.5 Expert Decomposition

Since we evaluated our results externally (using the MoJoFM), it was important to have reliable expert decompositions with which to compare our clustering results. For the proprietary software systems, expert decompositions were developed by personnel having design and development experience in the software industry. They had 6 to 7 years of experience in developing software systems using C++. Some of the experts were the original developers of the software systems. We provided the source code and class listing to all the experts and asked them to develop a decomposition of the given system. The experts were not provided with any details about clustering algorithms, and what relationships between entities were utilized during clustering. A summary of relevant statistics of the experts are presented in Table 32.

Table 32 Personnel Statistics

Personnel	System	Experience in Years
Actual designer	DDA	7
Experienced in C++	FES	6
Actual designer	PEDS	7
Maintainer	PLC	7
Actual designer	PLP	6
Experienced in C++	SAVT	7

The process of preparing the expert decomposition is described in detail in (Muhammad *et al.*, 2012) and (Naseem *et al.*, 2013). These expert decompositions have been previously used in different studies (Naseem *et al.*, 2013), (Muhammad *et al.*, 2012).

For Mozilla, we used the expert decomposition used in (Siddique and Maqbool, 2012). (Siddique and Maqbool, 2012) have taken this decomposition from Xia and Tzerpos (2005). For Weka, we used the expert decomposition given in (Patel *et al.*, 2009). This expert decomposition was provided by the original designers of the Weka software system. The expert decompositions for these systems have been used for modularization experiments earlier in (Siddique and Maqbool, 2012), (Patel *et al.*, 2009), (Andreopoulos and Tzerpos, 2005) and (Hussain *et al.*, 2015).

5 Assessment of the new similarity measure

In this section, we present the experimental results using arbitrary decisions, number of clusters, and authoritativeness using MoJoFM measure.

5.1 Arbitrary Decision

The overall results for all similarity measures are reported in Table 33. This table lists the average number of arbitrary decisions that are made by the AHC using different similarity measures in each iteration. The first column in Table 33 shows the methods, that is, CL, SL, and WL. Similarity measures are shown in the second column, while the arbitrary decision values for all test systems are given in the next eight columns. The second last column presents the average values for each similarity measure, and the last column shows the average values for new and existing measures. The bold face values enclosed in parentheses indicate best values, while only bold face values represent the better values for a certain system/method.

As can be seen from Table 33, our proposed similarity measures have reduced the arbitrary decisions for each method. It is very interesting that JCJNM, JCRR and JCJNMRR measures in most cases produce the same number of arbitrary decisions and also less than the existing JC and RR measures. JNMRR and JNM measures produce the same results except in two cases, where the difference is minor. Same results may be due to the fact that both similarity measures (JNM and RR) have the ability to count all features, that is, *a, b, c,* and *d*. Therefore, adding RR may have no additional effect on the similarity values of JNM to reduce arbitrary decisions.

Though our proposed measures produce a less number of arbitrary decisions, the existing JNM measure and our new JNMRR measure have the lowest number of arbitrary decisions. It is interesting to note that for the SL method, the JNMRR and JNM measures produce a less number of arbitrary decisions for all the test systems as compared to other methods.

It can be easily observed from the second last column of Table 33 that, as was expected, our proposed similarity measures produced a less number of arbitrary decisions on average, as compared to existing similarity measures (see the last column of Table 33).

To ease the analysis, we select the average values from the second last column of Table 33, and summarize them in the Table 34. Table 34 shows the average values of each similarity measure for linkage methods. It can be seen from second last column of Table 34 that for all methods on average, JNMRR and JNM produced a less number of arbitrary decisions. Meanwhile, last columns show the average values for all methods of our new and existing similarity measures separately in two rows. As can be seen, our new similarity measures on average produced 79.03 arbitrary decisions during the clustering process which are less than arbitrary decisions produced by existing similarity measures (86.44).

For further analysis, box-plots are used in Figure 3 to illustrate the arbitrary decisions made during the clustering process by AHC using different similarity measures. A box-plot shows the variation in the values of arbitrary decisions by indicating

Table 33 Experimental Results using Arbitrary Decisions for all Similarity Measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average	Average
CL	JCJNM	2.06	10.28	253.19	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	**129.69**
	JCRR	2.06	10.28	253.20	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	
	JNMRR	**1.89**	**10.26**	253.17	**9.80**	37.63	9.25	**18.89**	**695.13**	**129.50**	
	JCJNMRR	2.07	10.28	253.19	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	
	JC	4.34	10.43	**245.27**	10.95	72.10	10.72	30.19	700.39	135.55	139.86
	JNM	**1.89**	**10.26**	253.17	**9.80**	37.63	9.25	**18.89**	**695.13**	**129.50**	
	RR	33.79	11.93	296.01	14.78	50.79	27.39	31.52	769.93	154.52	
SL	JCJNM	1.55	3.00	4.07	**1.13**	**35.90**	**1.85**	**10.91**	374.69	54.14	**54.03**
	JCRR	1.55	3.00	4.08	**1.13**	**35.90**	**1.85**	**10.91**	374.69	54.14	
	JNMRR	**1.22**	**2.98**	**3.03**	**1.13**	**35.90**	**1.85**	**10.91**	**372.77**	**53.72**	
	JCJNMRR	1.55	3.00	4.05	**1.13**	**35.90**	1.86	**10.91**	374.69	54.14	
	JC	3.93	3.20	8.68	2.25	70.47	3.30	20.98	384.05	62.11	60.90
	JNM	**1.22**	**2.98**	**3.03**	**1.13**	**35.90**	**1.85**	**10.91**	**372.77**	**53.72**	
	RR	8.38	4.70	17.49	8.20	53.16	20.59	22.32	400.20	66.88	
WL	JCJNM	1.15	2.98	2.36	0.98	(**35.81**)	1.80	(**10.68**)	371.36	53.39	(**53.37**)
	JCRR	1.15	2.98	2.37	0.98	(**35.81**)	1.80	(**10.68**)	371.36	53.39	
	JNMRR	(**0.97**)	(**2.91**)	(**2.23**)	(**0.90**)	35.85	(**1.79**)	(**10.68**)	(**371.21**)	(**53.32**)	
	JCJNMRR	1.15	2.98	2.36	0.98	(**35.81**)	(**1.79**)	(**10.68**)	371.36	53.39	
	JC	3.06	3.07	4.43	1.93	70.28	3.20	20.30	374.15	60.05	58.55
	JNM	0.99	(**2.91**)	(**2.23**)	(**0.90**)	35.85	(**1.79**)	10.69	(**371.21**)	(**53.32**)	
	RR	4.97	3.72	7.42	3.73	48.71	19.61	16.48	393.58	62.28	

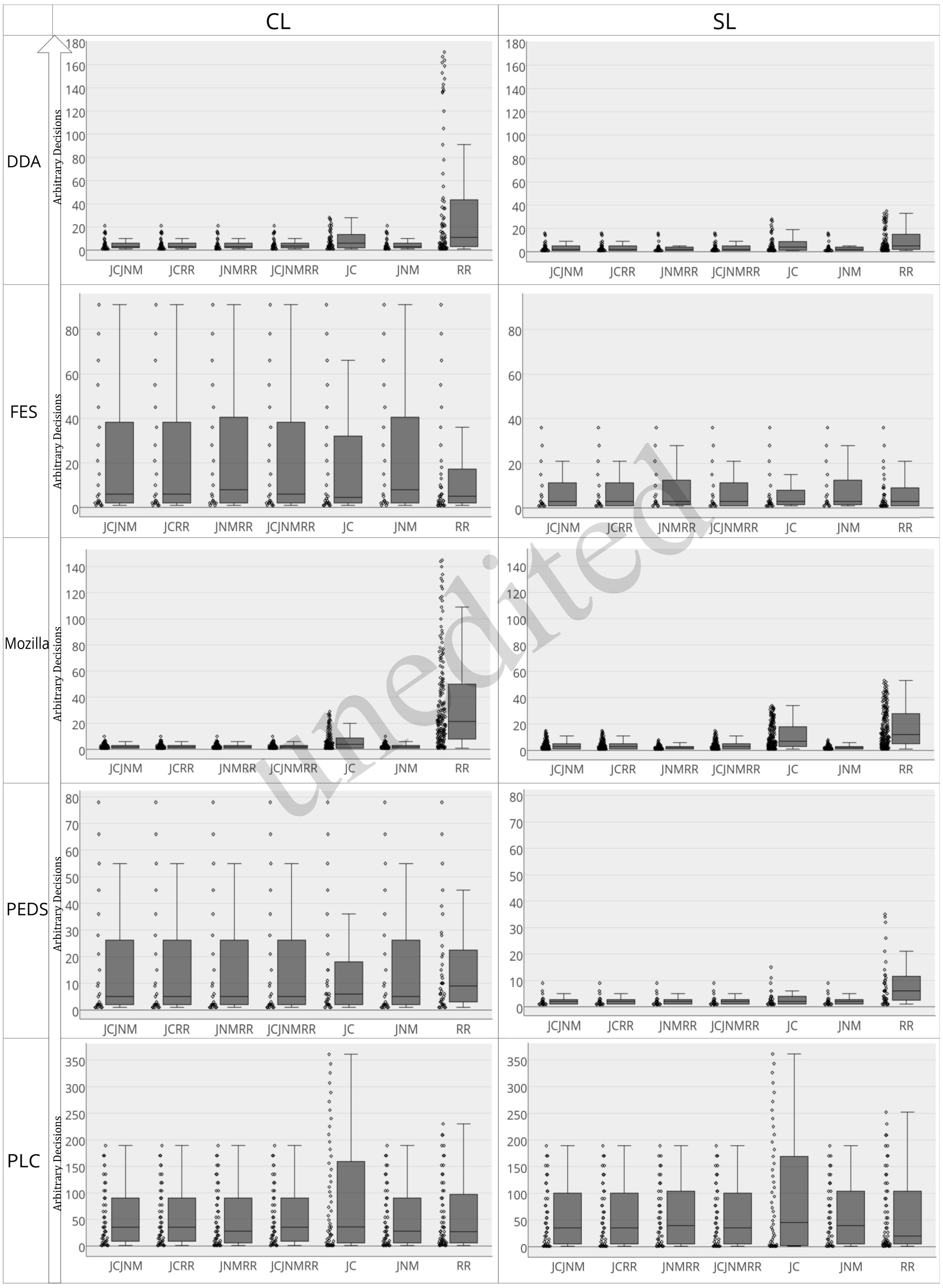

Fig. 3 Arbitrary decisions made during the clustering process by all binary similarity measures using CL and SL methods for different test software systems.

Table 34 Average Number of Arbitrary Decisions for all Similarity Measures

Measure	CL	SL	WL	Average	Average
JCJNM	129.76	54.14	53.39	79.09	**79.03**
JCRR	129.76	54.14	53.39	79.09	
JNMRR	129.50	53.72	53.32	**78.85**	
JCJNMRR	129.76	54.14	53.39	79.09	
JC	135.55	62.11	60.05	85.90	86.44
JNM	129.50	53.72	53.32	**78.85**	
RR	154.52	66.88	62.28	94.56	

the quartiles and also highlights the outliers for each measure. For clarity, points are presented alongside a box-plot, which represent the number of iterations in which that many arbitrary decisions made (i.e., the number of times that value of arbitrary decisions arises during clustering). Thus, the density of the points shows which arbitrary decision values are mostly observed during the clustering process.

As shown in Figure 3, JC and RR produce a larger number of arbitrary decisions as compared to JCJNM, JCRR, JNMRR, JCJNMRR and JNM in general. This is apparent from the height of the box, which indicates higher dispersion. Although in some cases the height of the box is lower for JC and RR (e.g., for FES using SL), there is a higher average in Table 33 for these measures. This indicates that arbitrary decisions are being made in a large number of iterations.

We also list some important statistics for our new and existing measures. Table 35 presents, for the arbitrary decisions, the minimum value (Min), maximum value (Max), mean, standard deviation (Std), Mode, and Percentage of the count of occurrence of mode value. The Min value for new measures is 78.85, which is equal to the Min value of existing measures. However, the Max value for new measures (79.09) is smaller than the existing measures. It indicates that new measures create the maximum, which is very near to the minimum value created by both the existing and new measures. It can be seen that the Std for our new similarity measures is 0.11, which is much less than the Std value for existing similarity measures. It clearly indicates that our new similarity measures produce very compact results as compared to existing similarity measures. It can also be observed that 75% of the results produced by our proposed measures are the same, while the existing measures created varied results and hence found no mode.

Table 35 Statistics of Arbitrary Decisions

Measure Type	Min	Max	Mean	Std	Mode	Percentage
New	78.85	79.09	79.03	0.11	79.09	75%
Existing	78.85	94.56	86.44	6.42	N/A	0%

5.2 Number of Clusters

The number of clusters during the clustering process shows how compact the produced clusters are. A large number of clusters created during the clustering process indicates that created clusters are highly compact, while a low number of clusters presents that created clusters are non-cohesive (Wang *et al.*, 2010) (Maqbool and Babri, 2007). Hence, a high number of clusters indicates the usefulness of the AHC approach.

Table 36 shows the maximum number of non-singleton clusters, created by AHC during all iterations. The values enclosed in parentheses indicate best values, while only bold face values represent the better values for a system/method. As can be seen from Table 36, the number of clusters created by AHC using our new similarity measures is higher than that created by existing similarity measures. It can be seen that for all test software systems, JCJNM, JCRR and JC measures created a large number of clusters using the CL, SL, and WL methods except for four cases, that is, CL applied on PLC, SL applied on DDA and Mozilla, and WL applied on Mozilla and Weka. For the SL method, JCJNM, JCRR, and JCJNMRR measures produced a large number of clusters for all test system except one (SAVT). It is interesting to note that once again, JNMRR and JNM measures result in the same number of clusters for all test software systems and linkage methods. It can also be seen that for Mozilla and Weka software systems, our new measures substantially increased the number of clusters similar to the JC measure. It is very interesting to note that the new measures integrating the JC similarity measure have achieved equally a large number of clusters as the JC measure. This is because our new measures (JCJNM, JCRR, and JCJNMRR) integrate the JC measure, which results in a large number of clusters. The JNMRR does not integrate the JC measure; therefore, results are the same as those of the JNM measure.

As can be seen from the second last column of Table 36, on average for the CL method, the JCJNM, JCRR, JCJNMRR, and JC produce better results, that is, they create a large numbers of clusters. The JCJNM and JCRR measures with the SL method and the JCJNMRR measure with the WL method result in a large number of clusters. It can be seen from the last column of Table 36 that, for each linkage method our new similarity measures outperform the existing one.

To summarize the results, we listed the values given in the second last column of Table 37 in Table 36. Table 37 presents the number of cluster values of similarity measures for linkage methods. This table clearly indicates how well a measure performs as compared to other contesting measures. This table also shows the average of each type, which indicates the average of all new measures and all existing measures separately. It can be seen that the new measures outperform the existing ones by creating a large number of clusters.

Table 37 Average Number of Clusters Produced during clustering for All Similarity Measures

Measure	CL	SL	WL	Average	Average
JCJNM	**26.88**	**17.38**	24.13	**22.79**	**20.90**
JCRR	**26.88**	**17.38**	24.13	**22.79**	
JNMRR	22.38	6.88	16.38	15.21	
JCJNMRR	**26.88**	17.25	**24.25**	**22.79**	
JC	**26.88**	17.25	24.13	22.75	16.97
JCNM	22.38	6.88	16.38	15.21	
RR	17.63	6.13	15.13	12.96	

For further analysis, a violin plot is shown in Figure 4, which shows the number of clusters created during the clustering process by different similarity measures using CL and SL on various test software systems[4]. The clustering process starts by forming very small-sized clusters (size 2). Increase in the height of violin graph means that a large number of clusters are created during the clustering process. The width of the violin shows the values distribution over the observed number of clusters. It can be easily observed that our new similarity measures and JC (in most of the cases) result in a large number of clusters. It can also be seen that JNM, RR, and JNMRR measures produce a relatively fewer number of clusters during the clustering process. This is especially true for the SL measure, where the number of clusters during clustering is much lower for these measures.

To support the claim that our proposed measures perform better, Table 38 shows the statistics of Table 37. All the statistical data (Min, Max, Mean, Std, Mode and Percentage) are derived from the second last column of Table 37. The Min value for new measures is 15.21, which is greater than the Min value of existing measures (12.96). The Max value for the new measures is slightly greater than the existing measures. It can be seen that the Std value for our proposed measures (3.28) is less than the existing measures, which indicates that our new measure's results are close to the mean value and are also close to each other. This table also shows the Mode and

[4]Due to the limited space only the results of CL and SL methods employed on DDA, FES, Mozilla, PEDS and PLC test software systems are presented

Table 36 Experimental Results using Number of Clusters Produced during Clustering for All Similarity Measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average	Average
CL	JCJNM	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	(**25.75**)
	JCRR	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	
	JNMRR	21	9	64	**12**	(**11**)	11	16	35	22.38	
	JCJNMRR	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	
	JC	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	22.29
	JNM	21	9	64	**12**	(**11**)	11	16	35	22.38	
	RR	17	8	49	10	10	9	14	24	17.63	
SL	JCJNM	**17**	**8**	**49**	**8**	**6**	**8**	12	**31**	**17.38**	**14.72**
	JCRR	**17**	**8**	**49**	**8**	**6**	**8**	12	**31**	**17.38**	
	JNMRR	7	4	11	5	5	4	7	12	6.88	
	JCJNMRR	**17**	**8**	48	**8**	**6**	**8**	12	**31**	17.25	
	JC	16	**8**	48	**8**	**6**	**8**	**13**	**31**	17.25	10.08
	JNM	7	4	11	5	5	4	7	12	6.88	
	RR	7	4	8	5	4	4	5	12	6.13	
WL	JCJNM	**20**	**9**	68	**12**	**9**	**11**	**17**	**47**	24.13	**22.22**
	JCRR	**20**	**9**	68	**12**	**9**	**11**	**17**	**47**	24.13	
	JNMRR	15	7	45	8	7	10	13	26	16.38	
	JCJNMRR	**20**	**9**	**69**	**12**	**9**	**11**	**17**	**47**	**24.25**	
	JC	**20**	**9**	**69**	**12**	**9**	**11**	**17**	46	24.13	18.54
	JNM	15	7	45	8	7	10	13	26	16.38	
	RR	15	7	41	8	8	9	11	22	15.13	

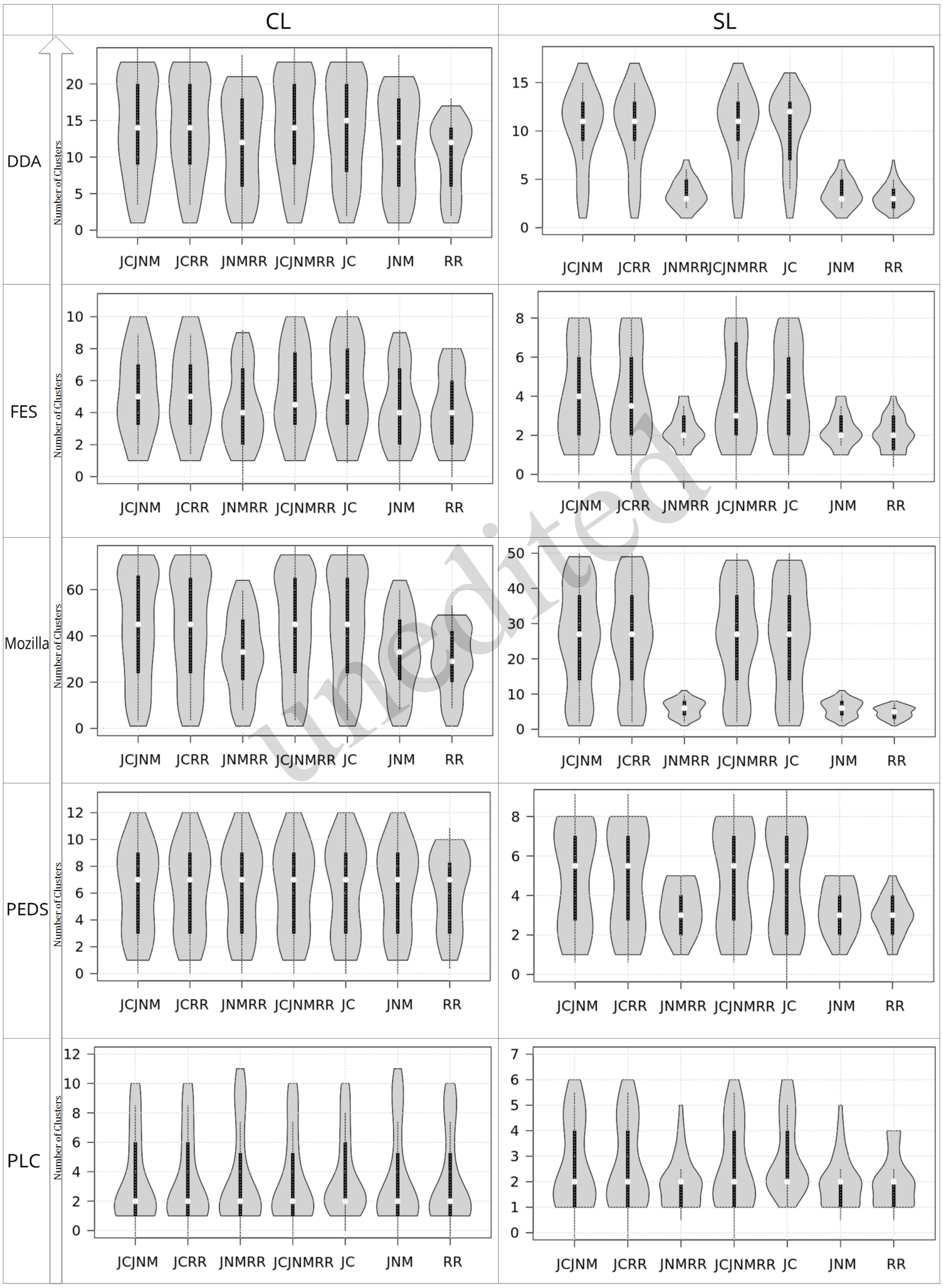

Fig. 4 Number of clusters created by all binary similarity measures during the clustering process using CL and SL methods for different test software systems.

Percentage of mode value's occurrence. The mode for new measures is 22.79 which occurs three times in the second column of Table 38. The percentage value 75 means that 75% values are the same for the JCJNM, JCRR and JCJNMRR similarity measures.

Table 38 Statistics of Number of Clusters Based on Table 37

Measure Type	Min	Max	Mean	Std	Mode	Percentage
New	15.21	22.79	20.90	3.28	22.79	75%
Existing	12.96	22.75	16.97	4.19	#N/A	0%

5.3 Authoritativeness

The automated result is required to approximate the decomposition prepared by a human expert (an authority). For this purpose we use MoJoFM to compare the automated results with the expert decompositions. The MoJoFM values for the series of experiments are given in Table 39. This table shows the maximum MoJoFM values selected during the iterations of the clustering process for all similarity measures and test software systems. The bold face values indicate the better values for a test system/method. The values enclosed in parentheses indicate best values in the Table 39. The average values for each similarity measure is shown in the second last column of Table 39, while the last column presents the average for new measures and existing measures and are based on the average values given in the second last column. As can be seen from Table 39, in most of the cases our new measures outperform the existing ones. This is because in previous Subsections 5.1 and 5.2, we have shown that our new similarity measures result in a smaller number of arbitrary decisions and a large number of clusters. Thus reducing arbitrary decisions and increasing the number of clusters improve the authoritativeness of the automated results (Naseem *et al.*, 2013) (Maqbool and Babri, 2007).

It can be easily analyzed that our new measures produce better results than the existing measures except for one test software system, that is, the PLC software system where results of the JNM measure using the CL method, is better. As can be seen from Table 39, for the DDA software system the JNMRR measure using the CL method results in the highest MoJoFM value. For the FES software system, the JCJNM, JCRR, and JCJNMRR measures using the CL method produce better results as compared to all other stratagems. For the Mozilla test system, the JNMRR similarity measure using the CL method gives the highest MoJoFM value. For the PEDS software system, our new and existing measures perform equally well. PLC is the only test systems for which the existing JNM measure using the CL method results in the highest MoJoFM value. For the PLP software system, all the existing measures using the CL method give the highest value, while for SAVT test software system JNMRR measure using the CL method and JCJNM and JCRR measures using the WL method result in the highest MoJoFM values (67.03%). Lastly, for the Weka software system, the JCJNM and JCRR measures using the WL method produce the highest MoJoFM values as compared to other measures using the CL and SL methods.

As can be seen from the second last column in Table 39, on average our new measures outperform the existing measures. The last column of Table 39 shows the average values of the new and existing similarity measures, respectively. As can be seen, our new measures outperform all existing measures using all linkage methods on all test software systems. Meanwhile, the CL method results in the highest MoJoFM value. Hence we can infer that the CL method produces better results as compared to the SL method, while the WL method falls in between these two.

To show results more precisely, Table 40 lists average values from the second last column of Table 39 for each similarity measure. It can be seen from Table 40, that for the CL method, the JNMRR measure results in better MoJoFM values as compared to other similarity measures. For the SL method, JCJNMRR measure yields better results, while for the WL method, the JCJNM and JCRR produce better results. As can be seen from the second last column of Table 40, on average, the JCJNMRR produces better result as compared to all other similarity measures. This table also provides good evidence to say that our new measures produce better results as compared to existing measures for all linkage methods.

Table 41 lists different statistics based on values in the second last column of Table 40. It can be seen that Min, Max, and Mean values for proposed new measures are higher than the existing measures. It indicates that in all cases, our new measures produce better results. These values indicate the superiority

Table 39 MoJoFM Results for all Similarity Measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average	Average
CL	JCJNM	56.25	**45.00**	63.89	**57.14**	61.54	(**65.67**)	**65.93**	**30.45**	55.73	(**55.94**)
	JCRR	56.25	**45.00**	63.89	**57.14**	61.54	**65.67**	**65.93**	**30.45**	55.73	
	JNMRR	(**60.00**)	**45.00**	**64.68**	**57.14**	61.54	**65.67**	(**67.03**)	30.13	(**56.40**)	
	JCJNMRR	57.50	**45.00**	63.89	**57.14**	61.54	**65.67**	**65.93**	**30.45**	55.89	
	JC	56.25	43.00	63.00	**57.14**	61.00	51.00	54.00	**30.45**	51.98	52.45
	JNM	56.25	43.00	64.00	**57.14**	(**65.00**)	60.00	54.00	30.13	53.69	
	RR	53.75	38.00	62.00	**57.14**	64.00	55.00	58.00	25.64	51.69	
SL	JCJNM	**53.75**	(**47.50**)	**46.03**	**57.14**	63.08	**59.70**	**58.24**	22.12	50.95	**47.97**
	JCRR	**53.75**	(**47.50**)	**46.03**	**57.14**	63.08	**59.70**	**58.24**	22.12	50.95	
	JNMRR	25.00	32.50	35.71	54.29	60.00	38.81	47.25	17.31	38.86	
	JCJNMRR	**53.75**	(**47.50**)	**46.03**	**57.14**	**64.62**	**59.70**	**58.24**	22.12	**51.14**	
	JC	**53.75**	35.00	46.00	54.29	55.00	28.00	32.00	**23.08**	40.89	36.16
	JNM	25.00	43.00	36.00	54.29	42.00	28.00	32.00	17.31	34.70	
	RR	23.75	33.00	36.00	51.43	42.00	28.00	32.00	16.99	32.90	
WL	JCJNM	**56.25**	42.50	61.11	**57.14**	**63.08**	**59.70**	(**67.03**)	(**32.05**)	**54.86**	**53.95**
	JCRR	**56.25**	42.50	61.11	**57.14**	**63.08**	**59.70**	(**67.03**)	(**32.05**)	**54.86**	
	JNMRR	50.00	40.00	56.35	54.29	**63.08**	58.21	63.74	24.68	51.29	
	JCJNMRR	**56.25**	42.50	**62.30**	**57.14**	**63.08**	**59.70**	(**67.03**)	30.45	54.81	
	JC	**56.25**	38.00	61.00	**57.14**	56.00	46.00	48.00	31.09	49.19	49.40
	JNM	50.00	35.00	61.00	54.29	55.00	55.00	53.00	24.68	48.50	
	RR	52.50	**45.00**	62.00	54.29	61.00	55.00	51.00	23.40	50.52	

of our new measures over existing measures. Standard deviation (Std) values for both type of measures indicates that each type has one value that is out of the range. Meanwhile, the JCJNM and JCRR measures result in exactly the same values, which can be seen in Table 40 and Table 41 (column Mode). The percentage value 50 means that 50% values are the same, that is, for the JCJNM and JCRR similarity measures.

Table 40 Average MoJoFM Results for all Similarity Measures

Measure	CL	SL	WL	Average	Average
JCJNM	55.73	50.95	**54.86**	53.85	**52.62**
JCRR	55.73	50.95	**54.86**	53.85	
JNMRR	**56.40**	38.86	51.29	48.85	
JCJNMRR	55.89	**51.14**	54.81	**53.94**	
JC	51.98	40.89	49.19	47.35	46.01
JNM	53.69	34.70	48.50	45.63	
RR	51.69	32.90	50.52	45.04	

Table 41 Statistics for New and Existing Measure based on Table 40

Measure Type	Min	Max	Mean	Std	Mode	Percentage
New	48.85	53.94	52.62	2.18	53.85	50%
Existing	45.04	47.35	46.01	0.98	#N/A	0%

5.3.1 Significance of the Authoritativeness

In order to show the significance of the authoritativeness' results, the t-test is conducted to show if there is significant difference between the MoJoFM values of new measures and the existing measures. In this case, null hypothesis H_o is as follows: "There is no difference between the MoJoFM values of new measures and the existing measures".

Table 42 The T-test Values between the MoJoFM Results of the New and Existing Measures using CL and SL Method

New Measures	Existing Measures	CL	SL
JCJNM	JC	1.90	2.41
	JNM	1.30	4.10
	RR	2.79	5.10
JCRR	JC	1.90	2.41
	JNM	1.30	4.10
	RR	2.79	5.10
JNMRR	JC	2.24	-0.45
	JNM	1.64	1.31
	RR	3.19	2.39
JCJNMRR	JC	2.01	2.46
	JNM	1.42	4.11
	RR	2.93	5.12

T-test (with $df = 7$) is conducted between the new and existing measures using CL and SL methods. The t-test results are given in Table 42. This table clearly indicates that if $\mathbf{t} > 2.365$, then the new measures have significantly performed better at 95% confidence level; at this level of confidence, all new measures are significantly better than RR using the CL method, and JCJNM, JCRR, and JCJNMRR are better than all existing measures using SL. Moreover, JNMRR is significantly better than RR using the SL method. At 90% confidence level ($\mathbf{t} > 1.895$), all new measures are significantly better than JC using the CL method. At 80% confidence level ($\mathbf{t} > 1.415$), JNMRR and JCJNMRR are significantly better than JNM using the CL method. Finally, the JCJNM and JCRR are significantly better than JNM using the CL method at 70% confidence level ($\mathbf{t} > 1.12$). Thus, the new measures produced significantly better results at different confidence levels; hence, the null hypothesis H_o is rejected.

5.4 Summary of Results

Considering the overall performance of the similarity measures, our new similarity measures (JCJNM, JCRR, JNMRR, and JCJNMRR) outperform the existing similarity measures (JC, JNM and RR) using the CL, SL and WL methods with respect to authoritativeness, number of clusters and arbitrary decisions. It can be noticed that for most of the cases the JCJNM, JCRR, and JCJNMRR measures yield better results. However, on average the JNMRR using the CL method results in the highest authoritativeness value as shown in Table 39. The JCJNM and JCRR measures used with the WL method achieve the second position, and the JCJNMRR measure using the SL method is on the third position. On average, the JCJNMRR measure gives better authoritativeness for all methods as compared to all other measures, which can be seen in Table 40.

Experimental results in terms of the number of clusters reveal that for all test software systems, the JCJNM, JCRR, JCJNMRR and JC measures using the CL method yield better results, as can be seen in Table 36. However, for all methods, the JCJNM measure performs better as shown in Table 37.

As can be seen from Table 33, the JNMRR measure using the WL method creates the lowest arbitrary decisions for all test software systems, as compared to other stratagems. For all methods, the JNMRR and JNM measures produce a less number of arbitrary decisions, as shown in Table 34.

As compared to the SL and WL methods, the CL method yield better results for authoritativeness for all the test software systems, as shown in Table 39. The WL method results in a less number of arbitrary decisions, as presented in Table 33. The CL method creates a large number of clusters during clustering as compared to the WL and SL methods, which can be seen in Table 36.

5.5 Threats to Validity

The selection of test systems may pose a threat to a study, since the test system characteristics may influence the results. To reduce this threat, we selected open and closed source software systems of different sizes (numbers of source lines) having different application domains. We used 8 test systems developed in well-known programming languages, that is, C, C++, and Java. However, more experiments may be conducted on large-size software systems developed in other programming languages.

In order to evaluate the results, we used MoJoFM as the external assessment criterion, and used the number of clusters and the number of arbitrary decisions as the internal assessment criteria. As MoJoFM requires an expert decomposition prepared by a human, a threat may exist during the preparation of expert decomposition due to human biases. However, we have tried to reduce this threat by selecting experienced software personnel as experts, who were also involved in the development of the proprietary software systems. For Mozilla, expert decomposition is prepared by the authors of (Godfrey and Lee, 2000) and then updated for the Mozilla 1.3 by the authors of (Xia and Tzerpos, 2005) and for Weka we used the decomposition prepared by the actual designers of the systems (Patel *et al.*, 2009). Moreover, we selected software systems that have been used in previous studies, with their expert decompositions (Siddique and Maqbool, 2012) (Muhammad *et al.*, 2012) (Patel *et al.*, 2009) (Andreopoulos and Tzerpos, 2005) (Hussain *et al.*, 2015). Since our main goal was to integrate the strengths of existing measures, our choices of internal assessment criteria are based on that goal. We are aware that there exist a number of other internal assessment criteria, for example, coupling and cohesion that can also be used to investigate the results.

6 Conclusions

This paper presents the improved binary similarity measures, namely, JCJNM, JCRR, JNMRR, and JCJNMRR for clustering software systems for modularization. These measures integrate the strengths of the following existing binary similarity measures: Jaccard (JC), JaccardNM (JNM), and Russell&Rao (RR). An example of the existing and new similarity measures is presented to show how our new measures integrate strengths of the existing similarity measures. We present empirical results to compare the results of the existing and new similarity measures and show the strengths of our new measures. The experimental results are obtained using eight real-world, different test software systems developed in different programming languages, and belonging to different domain applications.

The empirical results indicate that our new binary similarity measures are able to produce better authoritative results than those of the existing similarity measures. Our new measures reduce the arbitrary decisions and increase the number of clusters during the clustering process. The proposed similarity measures can be applied to test software systems developed in any programming language, since they depend only on binary feature vector representation of data.

It is possible to extend our integration approach to form new similarity and distance measures using existing similarity and distance measures for other clustering application domains.

References

Andreopoulos, Bill, Tzerpos, Vassilios, 2005. Multiple Layer Clustering of Large Software Systems. Working Conference on Reverse Engineering, p.79–88.

Andritsos, P., Tzerpos, Vassilios, 2005. Information-theoretic software clustering. *IEEE Transactions on Software Engineering*, **31**(2):150–165.

Anquetil, Nicolas, Lethbridge, Timothy C, 1999. Experiments with clustering as a software remodularization method. Working Conference on Reverse Engineering, p.235–255.

Bauer, Markus, Trifu, M., 2004. Architecture-aware adaptive clustering of OO systems. European Conference on Software Maintenance and Reengineering, p.3–14.

Bittencourt, Roberto Almeida, Guerrero, Dalton Dario Serey, 2009. Comparison of Graph Clustering Algorithms for Recovering Software Architecture Module Views. European Conference on Software Maintenance and Reengineering, p.251–254.

Cheetham, Alan H, Hazel, Joseph E, 1969. Binary (Presence-Absence) Similarity Coefficents. *Journal of Paleontology*, **43**(5):1130–1136.

Choi, Seung-Seok, Sung-Hyuk, Cha, Tappert, Charles C, 2010. A Survey of Binary Similarity and Distance Measures. *Journal of Systemics, Cybernetics & Informatics*, **8**(1):43–48.

Chong, Chun Yong, Lee, Sai Peck, Ling, Teck Chaw, 2013. Efficient software clustering technique using an adaptive and preventive dendrogram cutting approach. *Information and Software Technology (IST)*, **55**(11):1994–2012.

Cui, Jian Feng, Chae, Heung Seok, 2011. Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems. *Information and Software Technology (IST)*, **53**(6):601–614.

Davey, J., Burd, Elizabeth, 2000. Evaluating the suitability of data clustering for software remodularisation. Working Conference on Reverse Engineering, p.268–276.

Dugerdil, Philippe, Jossi, Sebastien, 2008. reverse-architecting legacy software based on roles: an industrial experiment. *Communications in Computer and Information Science*, **22**(Part 2):114–127.

Glorie, Marco, Zaidman, Andy, van Deursen, Arie, Hofland, Lennart, 2009. Splitting a large software repository for easing future software evolution-an industrial experience report. *Journal of Software Maintenance and Evolution: Research and Practice*, **21**(2):113–141.

Godfrey, Michael W, Lee, Eric HS, 2000. Secrets from the monster: Extracting MozillaâĂŹs software architecture. International Symposium on Constructing Software Engineering Tools, p.1–10.

Hall, Mathew, Walkinshaw, Neil, McMinn, Phil, 2012. Supervised software modularisation. IEEE International Conference on Software Maintenance (ICSM), p.472–481.

Hussain, Ibrar, Khanum, Aasia, Abbasi, Abdul Qudus, Javed, M Younus, 2015. A Novel Approach for Software Architecture Recovery Using Particle Swarm Optimization. *The International Arab Journal of Information Technology*, **12**(1):1–10.

Jackson, Donald A, Somers, Keith M, Harvey, Harold H, 1989. Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence? *American Naturalist*, **133**(3):436–453.

Jahnke, J.H., 2004. Reverse engineering software architecture using rough clusters. IEEE Annual Meeting of the Fuzzy Information, p.4–9 Vol.1.

Kanellopoulos, Yiannis, Antonellis, Panos, Tjortjis, Christos, Makris, Christos, 2007. k-Attractors: A Clustering Algorithm for Software Measurement Data Analysis. IEEE International Conference on Tools with Artificial Intelligence, p.358–365.

Lakhotia, Arun, 1997. A unified framework for expressing software subsystem classification techniques. *Journal of Systems and Software*, **36**(3):211–231.

Lesot, M-J., Rifqi, M., 2009. Similarity measures for binary and numerical data : a survey. *Int. J. Knowledge Engineering and Soft Data Paradigms*, **1**(1).

Lung, Chung-Horng, Zaman, Marzia, Nandi, Amit, 2004. Applications of clustering techniques to software partitioning, recovery and restructuring. *Journal of Systems and Software*, **73**(2):227–244.

Lutellier, Thibaud, Chollak, Devin, Garcia, Joshua, Tan, Lin, Rayside, Derek, Medvidovic, Nenad, Kroeger, Robert, 2015. Comparing Software Architecture Recovery Techniques Using Accurate Dependencies. IEEE International Conference on Software Engineering (ICSE), p.69–78.

Maqbool, O., Babri, H. A., 2004. The weighted combined algorithm: a linkage algorithm for software clustering. European Conference on Software Maintenance and Reengineering, p.15-24.

Maqbool, Onaiza, Babri, H.A., 2007. Hierarchical Clustering for Software Architecture Recovery. *IEEE Transactions on Software Engineering*, **33**(11):759–780.

Mitchell, Brian S., 2006. Clustering Software Systems to Identify Subsystem Structures (Technical Report).

Mitchell, Brian S., Mancoridis, Spiros, 2006. On the automatic modularization of software systems using the Bunch tool. *IEEE Transactions on Software Engineering*, **32**(3):193–208.

Muhammad, Siraj, Maqbool, Onaiza, Abbasi, Abdul Qudus, 2012. Evaluating relationship categories for clustering object-oriented software systems. *IET Software*, **6**(3):260.

Naseem, Rashid, Maqbool, Onaiza, Muhammad, Siraj, 2010. An Improved Similarity Measure for Binary Features in Software Clustering. International Conference on Computational Intelligence, Modelling and Simulation, p.111–116.

Naseem, Rashid, Maqbool, Onaiza, Muhammad, Siraj, 2011. Improved Similarity Measures for Software Clustering. European Conference on Software Maintenance and Reengineering, Pakistan, p.45–54.

Naseem, Rashid, Maqbool, Onaiza, Muhammad, Siraj, 2013. Cooperative clustering for software modularization. *Journal of Systems and Software (JSS)*, **86**(8):2045–2062.

Patel, Chiragkumar, Hamou-Lhadj, Abdelwahab, Rilling, Juergen, 2009. Software Clustering Using Dynamic Analysis and Static Dependencies. 2009 13th European Conference on Software Maintenance and Reengineering, p.27–36.

Praditwong, Kata, 2011. Solving software module clustering problem by evolutionary algorithms. International Joint Conference on Computer Science and Software Engineering (JCSSE), p.154–159.

Praditwong, Kata, Harman, Mark, Yao, Xin, 2011. Software Module Clustering as a Multi-Objective Search Problem. *IEEE Transactions on Software Engineering (TSE)*, **37**(2):264–282.

Saeed, M., Maqbool, Onaiza, Babri, H.A., Hassan, S.Z., Sarwar, S.M., 2003. Software clustering techniques and the use of combined algorithm. Seventh European Conference on Software Maintenance and Reengineering, p.301–306.

Sartipi, Kamran, Kontogiannis, Kostas, 2003. On modeling software architecture recovery as graph matching. International Conference on Software Maintenance, p.224–234.

Shah, Zubair, Naseem, Rashid, Orgun, MehmetA., Mahmood, Abdun Naser, Shahzad, Sara, 2013. Software Clustering Using Automated Feature Subset Selection. International Conference on Advanced Data Mining and Applications, **8347**:47–58.

Shtern, Mark, Tzerpos, Vassilios, 2010. On the Comparability of Software Clustering Algorithms. 2010 IEEE 18th International Conference on Program Comprehension, p.64–67.

Shtern, Mark, Tzerpos, Vassilios, 2012. Clustering Methodologies for Software Engineering. *Advances in Software Engineering (ASE)*, **2012**:1–18.

Shtern, Mark, Tzerpos, Vassilios, 2014. Methods for selecting and improving software clustering algorithms. *Software: Practice and Experience*, **44**(1):33–46.

Siddique, Faiza, Maqbool, Onaiza, 2012. Enhancing comprehensibility of software clustering results. *IET Software*, **6**(4):283.

Synytskyy, N., Holt, Richard C., Davis, I., 2005. Browsing Software Architectures With LSEdit. International Workshop on Program Comprehension, p.176–178.

Tonella, Paolo, 2001. Concept analysis for module restructuring. *IEEE Transactions on Software Engineering*, **27**(4):351–363.

Tzerpos, Vassilios, 2003. An optimal algorithm for MoJo distance. IEEE International Workshop on Program Comprehension, p.227–235.

Tzerpos, Vassilios, Holt, Richard C., 1999. MoJo: a distance metric for software clusterings. Working Conference on Reverse Engineering, p.187–193.

Tzerpos, Vassilios, Holt, Richard C., 2000. On the stability of software clustering algorithms. International Workshop on Program Comprehension, p.211–218.

Vasconcelos, Aline, Werner, Claudia, 2007. Architecture recovery and evaluation aiming at program understanding and reuse. Software Architectures, Components, and Applications, p.72–89.

Veal, Benjamin WG, 2011. Binary similarity measures and their applications in machine learning. PhD thesis, London, United Kingdom: London School of Economics.

Wang, Yuxin, Liu, Ping, Guo, He, Li, Han, Chen, Xin, 2010. Improved Hierarchical Clustering Algorithm for Software Architecture Recovery. International Conference on Intelligent Computing and Cognitive Informatics, p.247–250.

Wen, Zhihua, Tzerpos, Vassilios, 2004. An effectiveness measure for software clustering algorithms. Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004., p.194–203.

Wiggerts, T.A., 1997. Using clustering algorithms in legacy systems remodularization. Working Conference on Reverse Engineering, p.33–43.

Wu, Jingwei, Hassan, A.E., Holt, Richard C., 2005. Comparison of clustering algorithms in the context of software evolution. IEEE International Conference on Software Maintenance, p.525–535.

Xanthos, Spiros, Goodwin, North, 2006. Clustering Object-Oriented Software Systems using Spectral Graph Partitioning. *Urbana*, **51**(1):1–5.

Xia, Chenchen, Tzerpos, Vassilios, 2005. Software Clustering Based on Dynamic Dependencies. European Conference on Software Maintenance and Reengineering, p.124–133.

Appendix

A Propositions for JCRR

Proposition A.1 Let the range of JCRR be z, then

$$JCRR(Ei,Ej)=\begin{cases} z=2, & \text{if } a=P \\ 0<z<2, & \text{if } 0<a<P \\ z=0, & \text{if } a=0 \end{cases}$$

Proof. As JCRR is the combined function of four quantities a, b, c, and d, as shown in Equation 12. Substituting value of $s = a + b + c$.

$$JCRR(Ei,Ej)=\frac{a(2(a+b+c)+d)}{(a+b+c)((a+b+c)+d)} \tag{15}$$

If all features are present in feature vectors of Ei and Ej, that is, $b = c = d = 0$, and $a =$ P, then above equation reduces to

$$JCRR(Ei, Ej) = \frac{a(2(a))}{(a)((a))} = 2 \qquad (16)$$

Therefore, the maximum similarity value that JCRR can create is 2
Now,
if no common feature is present in feature vectors of Ei and Ej, that is, $a = 0$ and $b + c + d \geq 0$, then Equation 15 reduces to

$$JCRR(Ei, Ej) = \frac{0(2(0 + b + c) + d)}{(0 + b + c)((0 + b + c) + d)} = 0 \qquad (17)$$

Thus the minimum similarity value that JCRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of Ei and Ej, that is, if $a = x$ and $b = c = d = y$, where x, $y > 0$, then Equation 12 reduces to

$$JCRR(Ei, Ej) = \frac{x(2(x + y + y) + y)}{(x + y + y)((x + y + y) + y)}$$

Above equation simplifies to

$$JCRR(Ei, Ej) = \frac{2x^2 + 5xy}{x^2 + 5xy + 6y^2} \qquad (18)$$

Equation 18 results in between 0 and 2, if x, $y>0$, $\forall$ Ei, Ej $\in F$.

□

Proposition A.2 JCRR satisfies Definition 2.1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^p$ and range is $\mathbb{R}^+$.

Proof. JCRR is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated only using binary values in feature vector of entities as defined in Section 2.2. Hence, the domain of JCRR measure is $\{0,1\}^p$. Meanwhile, JCRR results in a real value, that is, $\mathbb{R}^+$ as proved in Proposition A.1.

$$JCRR(Ei, Ej) = \begin{cases} z > 0, & \text{if } a \geq 1 \\ z = 0, & \text{otherwise} \end{cases}$$

□

Proposition A.3 JCJNM fulfills the properties of Positivity and Symmetry of a similarity measure.

Proof. Positivity: It has been shown in the proof of Proposition A.1, that JCRR creates similarity value equal to or greater than 0, that is, JCRR(Ei,Ej) $\rightarrow \mathbb{R}^+$, $\forall$ Ei, Ej $\in F$.

Symmetry: It is obvious that

$$JCRR(Ei, Ej) = JCRR(Ej, Ei)$$

□

Proposition A.4 JCRR fulfills the property of Maximality of a similarity measure.

Proof. Let us suppose that $b + c = x$ and x is a positive number; then Equation 12 of JCRR becomes

$$JCRR(Ei, Ej) = \frac{a(2(a + x) + d)}{(a + x)((a + x) + d)}$$

The above equation simplifies to

$$JCRR(Ei, Ej) = \frac{a(2a + 2x + d)}{a(a + d + 2x) + dx + x^2} \qquad (19)$$

To calculate similarity of an entity with itself, that is, JCRR(Ei,Ei) then it is sure that $x = 0$, a, $d \geq 0$. Using these quantities, Equation 19 of JCRR reduces to:

$$JCRR(Ei, Ei) = \frac{a(2a + d)}{a(a + d)} \qquad (20)$$

Therefore, using Equations 20 and 19, $\forall$ Ei, Ej $\in F$, the following association will always be true for any values of a, d, and x, where $a + d + x =$ P:

$$\frac{a(2a + d)}{a(a + d)} \geq \frac{a(2a + 2x + d)}{a(a + d + 2x) + dx + x^2}$$

□

B Propositions for JNMRR

Proposition B.1 Let the range of JNMRR be z, then

$$JNMRR(Ei, Ej) = \begin{cases} z = 1.5, & \text{if } a = P \\ 0 < z < 1.5, & \text{if } 0 < a < P \\ z = 0, & \text{if } a = 0 \end{cases}$$

Proof. As JNMRR is the combined function of four quantities *a, b, c*, and *d*, as shown in Equation 13, substituting value of $s = a + b + c$.

$$= \frac{a(3(a+b+c)+2d)}{(2(a+b+c)+d)((a+b+c)+d)} \tag{21}$$

If all features are present in feature vectors of Ei and Ej, that is, $b = c = d = 0$, and $a =$ P, then the above equation reduces to

$$JNMRR(Ei, Ej) = \frac{a(3(a))}{2(a)((a))} = 1.5 \tag{22}$$

Therefore, the maximum similarity value that JNMRR can create is 1.5
Now,
if no common feature is present in feature vectors of Ei and Ej, that is, $a = 0$ and $b + c + d \geq 0$, then Equation 21 reduces to

$$= \frac{0(3(0+b+c)+2d)}{(2(0+b+c)+d)((0+b+c)+d)} = 0 \tag{23}$$

Thus the minimum similarity value that JNMRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of Ei and Ej, that is, if $a = x$ and $b = c = d = y$, where x, $y > 0$, then Equation 12 reduces to

$$= \frac{x(3(x+y+y)+2y)}{(2(x+y+y)+y)((x+y+y)+y)}$$

The above equation simplifies to

$$JNMRR(Ei, Ej) = \frac{3x^2+8xy}{2x^2+11xy+15y^2} \tag{24}$$

Equation 24 results in between 0 and 1.5, if x, y>0, ∀ Ei, Ej ∈ F.

□

Proposition B.2 JNMRR satisfies Definition 2.1 given in Section 2.2, which states that domain of a binary similarity measure is $\{0,1\}^p$ and range is $\mathbb{R}^+$.

Proof. JNMRR is the combined function of four quantities *a, b, c,* and *d*, and all these quantities can be calculated only using binary values in feature vector of entities as defined in Section 2.2. Hence, the domain of JNMRR measure is $\{0,1\}^p$. Meanwhile, JNMRR results in a real value, that is, $z = \mathbb{R}^+$ as proved in Proposition B.1.

$$JNMRR(Ei, Ej) = \begin{cases} z > 0, & \text{if } a \geq 1 \\ z = 0, & \text{otherwise} \end{cases}$$

□

Proposition B.3 JNMRR fulfills the properties of Positivity and Symmetry of a similarity measure.

Proof. Positivity: It has been shown in the proof of Proposition B.1, that JNMRR creates similarity value equal to or greater than 0, that is, JNMRR(Ei,Ej) → $\mathbb{R}^+$, ∀ Ei, Ej ∈ F.

Symmetry: It is obvious that

$$JNMRR(Ei, Ej) = JNMRR(Ej, Ei)$$

□

Proposition B.4 JNMRR fulfills the Maximality property of a similarity measure.

Proof. Let us suppose that $b + c = x$ and x is a positive number, then Equation 12 of JNMRR becomes

$$JNMRR(Ei, Ej) = \frac{a(3(a+x)+2d)}{(2(a+x)+d)((a+x)+d)}$$

The above equation simplifies to

$$JNMRR(Ei, Ej) = \frac{a(3a+3x+2d)}{a(2a+3d+4x)+d^2+3dx+2x^2} \tag{25}$$

To calculate the similarity of an entity with itself, that is, JNMRR(Ei,Ei) then it is sure that $x = 0$, a, $d \geq 0$. Using these quantities, Equation 25 of JNMRR reduces to:

$$JNMRR(Ei, Ei) = \frac{a(3a+2d)}{a(2a+3d)+d^2} \tag{26}$$

Therefore, using Equations 26 and 25, ∀ Ei, Ej ∈ F, the following association will always be true for any values of *a, d*, and *x*, where $a + d + x =$ P:

$$\frac{a(3a+2d)}{a(2a+3d)+d^2} \geq \frac{a(3a+3x+2d)}{a(2a+3d+4x)+d^2+3dx+2x^2}$$

□

C Propositions for JCJNMRR

Proposition C.1 Let the range of JCJNMRR be z, then

$$JCJNMRR(Ei, Ej) = \begin{cases} z = 2.5, & \text{if } a = P \\ 0 < z < 2.5, & \text{if } 0 < a < P \\ z = 0, & \text{if } a = 0 \end{cases}$$

Proof. As JCJNMRR is the combined function of four quantities *a, b, c*, and *d*, as shown in Equation 14, substituting value of $s = a + b + c$.

$$= \frac{a(5(a+b+c)^2 + 5(a+b+c)d + d^2)}{(a+b+c)(2(a+b+c)^2 + 3(a+b+c)d + d^2)} \tag{27}$$

If all features are present in feature vectors of Ei and Ej, that is, $b = c = d = 0$, and $a =$ P, then the above equation reduces to

$$JCJNMRR(Ei, Ej) = \frac{a(5(a^2))}{(a)(2a^2)} = 2.5 \tag{28}$$

Therefore, the maximum similarity value that JCJNMRR can create is 2.5
Now,
if no common feature is present in feature vectors of Ei and Ej, that is, $a = 0$ and $b + c + d \geq 0$, then Equation 27 reduces to

$$= \frac{0(5(0+b+c)^2 + 5(0+b+c)d + d^2)}{(0+b+c)(2(0+b+c)^2 + 3(0+b+c)d + d^2)} = 0 \tag{29}$$

Thus the minimum similarity value that JCJNMRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of Ei and Ej, that is, if $a = x$ and $b = c = d = y$, where $x, y > 0$, then Equation 12 reduces to

$$= \frac{x(5(x+y+y)^2 + 5(x+y+y)y + y^2)}{(x+y+y)(2(x+y+y)^2 + 3(x+y+y)y + y^2)}$$

The above equation simplifies to

$$= \frac{5x^3 + 25x^2y + 31xy^2}{2x^3 + 15x^2y + 37xy^2 + 30y^3} \tag{30}$$

Equation 30 results in between 0 and 2.5, if x, $y>0$, $\forall$ Ei, Ej $\in F$.

$\square$

Proposition C.2 JCJNMRR satisfies Definition 2.1 given in Section 2.2, which states that domain of a binary similarity measure is $\{0,1\}^p$ and range is $\mathbb{R}^+$.

Proof. JCJNMRR is the combined function of four quantities *a, b, c,* and *d*, and all these quantities can be calculated only using binary values in feature vector of entities as defined in Section 2.2. Hence, the domain of JCJNMRR measure is $\{0,1\}^p$. Meanwhile, JCJNMRR results in a real value, that is, $z = \mathbb{R}^+$ as proved in Proposition C.1.

$$JCJNMRR(Ei, Ej) = \begin{cases} z > 0, & \text{if } a \geq 1 \\ z = 0, & \text{otherwise} \end{cases}$$

$\square$

Proposition C.3 JCJNMRR fulfills the properties of Positivity and Symmetry of a similarity measure.

Proof. Positivity: It has been shown in the proof of Proposition C.1, that JCJNMRR creates similarity value equal to or greater than 0, that is, JCJNMRR(Ei,Ej) $\rightarrow \mathbb{R}^+$, $\forall$ Ei, Ej $\in F$.

Symmetry: It is obvious that

$$JCJNMRR(Ei, Ej) = JCJNMRR(Ej, Ei)$$

$\square$

Proposition C.4 JCJNMRR fulfills the Maximality property of a similarity measure.

Proof. Let us suppose that $b + c = x$ and x is a positive number, then the Equation 12 of JCJNMRR becomes

$$= \frac{a(5(a+x)^2 + 5(a+x)d + d^2)}{(a+x)(2(a+x)^2 + 3(a+x)d + d^2)} \tag{31}$$

To calculate the similarity of an entity with itself, that is, JCJNMRR(Ei,Ei), then it is sure that $x = 0$, a, $d \geq 0$. Using these quantities, Equation 31 of JCJNMRR reduces to:

$$JCJNMRR(Ei, Ei) = \frac{a(5(a)^2 + 5(a)d + d^2)}{(a)(2(a)^2 + 3(a)d + d^2)}$$

The above equation simplifies to

$$JCJNMRR(Ei, Ei) = \frac{5a^2 + 5ad + d^2}{2a^2 + 3ad + d^2} \tag{32}$$

Therefore, using Equations 32 and 31, $\forall$ Ei, Ej $\in F$, the following association will always be true for any values of *a, d*, and *x*, where $a + d + x =$ P:

$$\frac{5a^2 + 5ad + d^2}{2a^2 + 3ad + d^2} \geq \frac{a(5(a+x)^2 + 5(a+x)d + d^2)}{(a+x)(2(a+x)^2 + 3(a+x)d + d^2)}$$

$\square$